Autonomous Agents and Multiagent Systems

2006/2007 – LEIC, IST

Lab 5 – Introduction to JADE

1 Objectives

- Introduction to the JADE platform

2 Hello World

So as to familiarize yourself with JADE, analyze the simple agent provided with the lab materials.

- Run `hello.bat`.
- Analyze the console output.
- Analyze the contents of the executable file.
- Analyze the agent code (`aasm.jade.HelloWorldAgent` in the `source` directory).
- JADE provides several graphical support tools. Run `hello-gui.bat` and explore these tools.

3 Behaviour

An agent in JADE can have several behaviours associated. A behaviour is defined using the class `jade.core.behaviours.Behaviour`. This classs defines two main methods: `action()` and `done()`. The first one implements the action of the behavior everytime it is exectured; the second one determines when the behaviour should end.

- Add to the `HelloWorldAgent` the internal class MyFirstBehaviour that implements a simple behaviour. This behavior ends after the first execution and its action is to write the sentence *"This is a simple behaviour!"* on screen. Associate the behavior to the agent by using the `addBehaviour` method.
- Extend this behaviour so that it receives in the constructor the number of times that the sentence should be written. This value should be passed to the agent in its construction modifying the file `hello.bat`.

JADE makes available several types of behaviours, including:

- `CyclicBehaviour` that represents an endless behaviour that runs in every execution cycle. In this class the method `done()` is redefined to always return f`alse`.
- `TickerBehaviour` that presents periodic behavior. In this class the `done` method is also redefined to allways return `false` and the `action` method is redefined to execute the method `onTick()`. The time interval must be specified in the instance constructor.
- `WakerBehaviour that represents behaviours that are executed, one, after a given period of time.` In this class, the `done()` method is redefined to return always `true` and the `action` method is redefined to execute the method `onWake()`. The waiting time is specified in the behaviour constructur. It is possible to run the behaviour again by calling the `reset()` method.

4 Communication

JADE includes a platform of agent communication. ACL (*Agent Communication Language*), based on the FIPA (*Foundation for Intelligent Physical Agents*) specification, is used as the communication language. Each message is represented by an instance of the class `jade.lang.acl.ACLMessage` that contains, amongst others, information about the type of message (*performative*), its content and sender and receiver. The communication is managed by the methods `send()` e `receive()`.

- Implement an Emissor agent that besides of writting the sentence "I'm sending a message to the *Receiver"* to the screen, sends a message to the mentioned *Receiver*. You should specify the type of the message in the message constructor. As example, use `ACLMessage.INFORM`. The name of the receptor should be received as an argument in the agent constructor (change the file `hello.bat`) and should correspond to the name of an agent that exists (this means it should be declared in the same file)
- Implement another agent `Receptor` that has a cyclic behaviour that verifies if the agent received any message and prints it on screen. The agent should wait for a new message. In order to do so, you should use the methods `receive() and block()`.
- Once, the previous agents are implemented, run `hello-gui.bat again`. Send some messages to a receptor and check if they are received.

In the above example, the receptor agent receives any kind of message. Nevertheless, this is not always desired. So, JADE allows us to filter received messages using an instance of the class `jade.lang.acl.MessageTemplate`.

- Add to the agent the behaviour that only process messages of the `ACLMessage.INFORM`. type. To select messages use the method `MatchPerformative` from the `MessageTemplate` class.
- Using the graphic interface, check the new agent behaviour to see if it only process the `INFORM` messages.
- There are additional filters. For instance the `MatchSender` method that selects messages according to the sender. Additionally it allows filter composition through the methods `and` and `or` from `MessageTemplate class`.

5 Case Study: Auctions

As described in the theory lessons, [Wooldridge02, Cap. 7], there are several kinds of auctions. In this lab two shall be simulated. For this purpose, consider the following entities:

- **Representative**: participates in an auction representing some user which created it and specified a maximum value to offer for a particular item also defined on creation;
- **Auctioneer**: coordinates the auction, receiving offers and selling items to the representatives with the best offers. Furthermore, inform the representatives whether their offers have been accepted. In the *English* auction, is also responsible for informing representatives whose offers where surpassed, so that they can update their offers.

a) *First-price sealed-bid* auction

In this auction, interested parties make one (and final) offer during a specified time period. When time finishes, the auctioneer chooses the best offer (typically, with the highest value) which is paid accordingly.

Explore the base code and implement this auction.

The following (incomplete) classes are provided:

```
aasm.jade.firstPriceSealedBidAuction.Auctioneer
aasm.jade.firstPriceSealedBidAuction.Representative
```

b) Yellow Pages

Note that the `getAuctioneers` method in the `Representative` class assumes that all auctioneers are "well known".

JADE supports the Yellow Pages mechanism (*Directory Facilitator*, DF). Explore this mechanism and alter the classes accordingly so that auctioneers are dynamically selected through the DF.

c) *English* auction

This auction proceeds in several rounds. First, every interested party makes a proposal. Then, the auctioneer, when the round time expires, evaluates the proposals and informs the participants about the highest offer.

A new round follows, where participants, whose offer is not the highest, can, if they so desire, make a new offer.

This auction may end in three cases: (1) when no new offers are made in a new round, i.e., no one makes a better offer than the one in the previous round; (2) when, only one offer is made, being this automatically the winner; (3) when maximum number of rounds is exceeded (this value is passed as an argument to the agent).

The auctioneer may have a base price for the item, which should be advertised at the beginning of the auction.